The invention claimed is:

1. A method of establishing a loop of delayed commands for completion among a larger set of commands in a queue of commands, comprising:

recognizing a delay in completion of a first read command in the queue and in response designating the first read command as a first delayed read command;

determining that the command in the queue following the first delayed read command is a second read command;

setting a loop start pointer to identify the first delayed read command;

setting a loop end pointer to identify the second read command;

recognizing a delay in completion of the second read command in the queue and in response designating the second read command as a second delayed read command;

determining that the command in the queue following the second delayed read command is a third read command;

advancing the loop end pointer to identify the third read command; and

attempting to complete the read commands at and between the loop start pointer and the loop end pointer until all of those read commands have been completed, before attempting to complete other commands in the queue.

2. A method as defined in claim 1, further comprising:

completing the first delayed read command; and

advancing the loop start pointer to identify the second delayed read command after the first delayed read command has been completed.

3. A method as defined in claim 2, further comprising:

recognizing a delay in completion of the third read command and in response designating the third read command as a third delayed read command;

determining that the command in the queue following the third read command is a fourth read command; and

advancing the loop end pointer to identify the fourth read command.

4. A method as defined in claim 2, further comprising:

recognizing the completion of all but one of the read commands between the loop start pointer and the loop end pointer; and

eliminating the loop start pointer and the loop end pointer upon recognizing the completion of all but the one read command.

5. A method as defined in claim 2 wherein the commands in the queue are selected for completion by a queue pointer, and the method further comprises:

adjusting the position of a queue pointer to move through the commands at and between the loop start pointer and the loop end pointer until all of the commands at and between the loop start pointer and the loop end pointer have been completed.

6. A method as defined in claim 5, further comprising:

limiting the position of the queue pointer to the commands at and between the loop start pointer and the loop end pointer until all of the commands at and between the loop start pointer and the loop end pointer have been completed.

7. A method as defined in claim 6, further comprising:

advancing the queue pointer in the sequence of commands starting with the command identified by the loop start pointer and continuing through all of the commands at and between the loop start pointer and the loop end pointer and ending with the command identified by the loop end pointer.

8. A method as defined in claim 7, further comprising:

marking each command identified at and between the loop start pointer and the loop end pointer as invalid upon completion of each such command.

9. A method as defined in claim 2, further comprising:

advancing the loop start pointer to identify the same command that is identified by the loop end pointer.

10. A method as defined in claim 9, further comprising:

abolishing the loop of delayed read commands by eliminating the loop start pointer and the loop end pointer upon loop start pointer identifying the same command that is identified by the loop end pointer.

11. A method as defined in claim 1, further comprising:

advancing the loop start pointer to the next subsequent command in the queue after completing the command identified by the loop start pointer.

12. A method as defined in claim 1, further comprising:

advancing the loop end pointer to the next subsequent read command in the queue upon experiencing a delay in completing the read command previously identified by the loop end pointer.

13. An interface controller for communicating commands and data to at least one target device attached to the interface controller; the interface controller comprising:

a command queue having a command buffer in which the commands are recorded;

a data mover for transferring commands to a target device and receiving responses from the target device in response to commands; and

a queue processor connected to the command queue and the data mover, the queue processor operatively applying selected commands from the command buffer to the data mover to be transferred to a target device for completion, the queue processor including programmed logic functionality which:

applies a first read command from the command buffer to the data mover for transfer to a target device for completion;

recognizes a response from the target device indicating a delay in completion of the first read command;

determines that the command in the command buffer following the first read command is a second read command;

sets a loop start pointer to identify the first read command;

sets a loop end pointer to identify the second read command;

applies the second read command from the command buffer to the data mover for transfer to a target device for completion;

recognizes a response from the target device indicating a delay in completion of the second read command;

determines that the command in the command buffer following the second read command is a third read command;

advances the loop end pointer to identify the third read command; and

applies the third read command from the command buffer to the data mover for transfer to a target device for completion.

14. An interface controller as defined in claim 13, wherein the queue processor includes further programmed logic functionality which:

identifies the read commands identified by and between the loop start pointer and the loop end pointer as delayed read commands;

establishes a loop of the delayed read commands; and

applies the delayed read commands of the loop to the data mover for transfer to the target device for completion before applying any other read commands from the command buffer to the data mover for transfer to the target device.

15. An interface controller as defined in claim 14, wherein the queue processor includes further programmed logic functionality which:

recognizes responses from each target device indicating completion of a read command transferred to the target device;

recognizes completion of the delayed read command identified by the loop start pointer; and thereafter

advances the loop start pointer to identify the next uncompleted delayed read command of the loop.

16. An interface controller as defined in claim 15, wherein the queue processor includes further programmed logic functionality which:

advances the loop start pointer to identify the delayed read command which is also identified by the loop end pointer upon recognizing that all of the other delayed read commands in the loop have been completed; and thereafter

abolishes the loop by eliminating the loop start pointer and the loop end pointer.

17. An interface controller as defined in claim 14, wherein the queue processor includes further programmed logic functionality which:

recognizes the completion of all but a remaining one of the delayed read commands in the loop; and thereafter

abolishes the loop by eliminating the loop start pointer and the loop end pointer.

18. An interface controller as defined in claim 17, wherein the queue processor includes further programmed logic functionality which:

recognizes a response from the target device indicating a delay in completion of the remaining one delayed read command; and thereafter

applies another command from the command buffer to the data mover for transfer to the target device for completion, the another command being a command not included in the loop before the loop was abolished.

19. An interface controller as defined in claim 17, wherein the queue processor includes further programmed logic functionality which:

applies the delayed read commands of the loop to the data mover for transfer to the target device for completion until the loop of delayed read commands is abolished, before applying any other read commands from the command buffer to the data mover for transfer to the target device.

20. An interface controller as defined in claim 14, wherein the commands of the command buffer are normally selected for completion by queue pointer positioning functionality of the interface controller which positions a queue pointer to identify the command of the command buffer selected for completion, and

wherein the queue processor further includes programmed logic functionality which:

overrides the normal positioning of the queue pointer positioning functionality of the interface controller to control the positioning of the queue pointer when the loop has been established.

21. An interface controller as defined in claim 13, wherein the commands of the command buffer are normally selected for completion by queue positioning functionality of the interface controller which positions a queue pointer to identify the command of the command buffer selected for completion, and the queue processor includes further programmed logic functionality which:

overrides the normal positioning of the queue positioning functionality of the interface controller; and

limits the position of the queue pointer to those commands identified by and between the loop start pointer and the loop end pointer until all of the uncompleted read commands identified between the loop start pointer and the loop end pointer have been completed.

22. An interface controller as defined in claim 21, wherein the queue processor includes further programmed logic functionality which:

marks each delayed read command identified by and between the loop start pointer and the loop end pointer as invalid upon completion of that command.

23. An interface controller as defined in claim 13, wherein the queue processor includes further programmed logic functionality which:

advances the loop start pointer to the next subsequent command in the command buffer after the command identified by the loop start pointer has been completed.

24. An interface controller as defined in claim 13, wherein the queue processor includes further programmed logic functionality which:

advances the loop end pointer to the next subsequent read command in the command buffer upon recognizing the first delay in completing the preceding read command.

25. An interface controller as defined in claim 13, wherein the queue processor includes further programmed logic functionality which:

determines that the command in the command buffer following the third read command is a fourth read command;

advances the loop end pointer to identify the fourth read command; and

applies the fourth read command from the command buffer to the data mover for transfer to a target device for completion.